Automated Line Labelling: Dataset for Contour Detection and 3D Reconstruction

Supplementary Material

Hari Santhanam* Nehal Doiphode* Jianbo Shi
University of Pennsylvania
{harisan, lahen, jshi}@seas.upenn.edu

1. Huffman-Clowes Labelling

As shown in Section 3.1, the Huffman-Clowes [5][4] line labelling scheme results in three types of contours: convex, concave, and obscuring. In Fig. 1, we show example 2D scenes with the convexity marked, to provide a visual for the definitions provided in Section 3.1. We do not mark every contour, but provide example labels for each contour type. Note again that the obscuring contour type is the combination of limb and occluding contours. Understanding the different contour types is crucial for line labelling.

2. Contour Extraction: Occlusion Cases

In this section, we describe the 4 major cases of occlusion, through visual examples, and expand on the details in Section 3.2. Moreover, we explain the algorithmic details further. Recall that occlusion detection required a forward and backward search. Again, for a contour C, in the forward search let $\hat{c_{1f}}$ be the first visible point and $\hat{c_{2f}}$ be the last visible point. In the backward search, let $\hat{c_{1b}}$ be the first visible point and $\hat{c_{2b}}$ be the last visible point. In Fig. 2, we show the main occlusion cases. In the leftmost image, the contour C is completely occluded, yet c_1 and c_2 are visible. The resulting forward and backward search yields case (1): $\hat{c_{1f}} = \hat{c_{2f}} = c_1$ and $\hat{c_{1b}} = \hat{c_{2b}} = c_2$. In the middle image, the contour C is partially occluded, specifically the portion containing c_2. The forward and backward search would yield case (2): $\hat{c_{1f}} = c_1$ and $\hat{c_{2f}} \neq c_2$, and c_2 is not visible. Case (3) is just the opposite of case (2) and would have a very similar contour occlusion. Finally, in the rightmost image, occlusion occurs somewhere between the endpoints. The forward and backward search would yield case (4): $\hat{c_{1f}} = c_1$, $\hat{c_{2f}} \neq c_2$, and $\hat{c_{1b}} = c_2$, $\hat{c_{2b}} \neq c_1$. These are the main types of occlusion that occur in our dataset.

*Equal contribution. Ordering determined at random.

3. Contour Classification Implementation

In section 3.3, we illustrate the process of contour classification, for obscuring, concave, and convex contours. Here, we present the details in the pseudocode shown in Alg. 1.

Data: $\{C_k\}$
Result: $contour\_type$ indicating the classification of contour C_k.
$tri1, tri2 \leftarrow$ triangles(C_k); //triangles form C_k
$type1 \leftarrow$ triangle_type($tri1$);
$type2 \leftarrow$ triangle_type($tri2$);
if $type1$ = *'visible' or* $type2$ = *'visible'* **then**
  return 'obscuring';
end
$c1, c2 \leftarrow$ centroid($tri1$), centroid($tri2$);
$n1, n2 \leftarrow$ normal($tri1$), normal($tri2$);
$d \leftarrow c1 - c2$;
$d \leftarrow d/(\|d\|$ + 1e-10);
$convexity \leftarrow n1 \cdot d - n2 \cdot d$;
if $convexity \geq 0$ **then**
  return 'convex';
else
  return 'concave';
end

Algorithm 1: Pseudocode for contour classification, given extracted visible contour C_k. We describe the function triangle_type in detail in Section 3.3. The Extended Convexity Criterion from [3], is written here as well.

4. Contour Grouping Implementation

In section 3.4, we show the process of contour grouping. Here, we present the intricacies with the pseudocode shown in Alg.2.

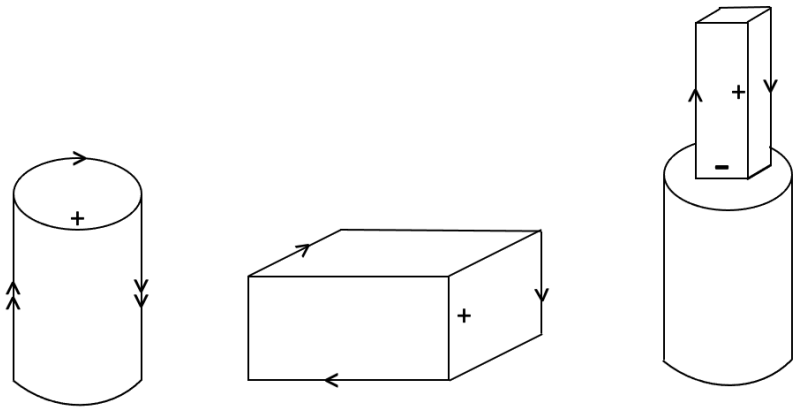

Figure 1. Examples of labelled Huffman-Clowes scenes [5][4][6]. '+' is convex, '-' is concave, '>' is 'occluding', and '>>' is limb. In our paper, occluding and limb contours combine to form the category obscuring.

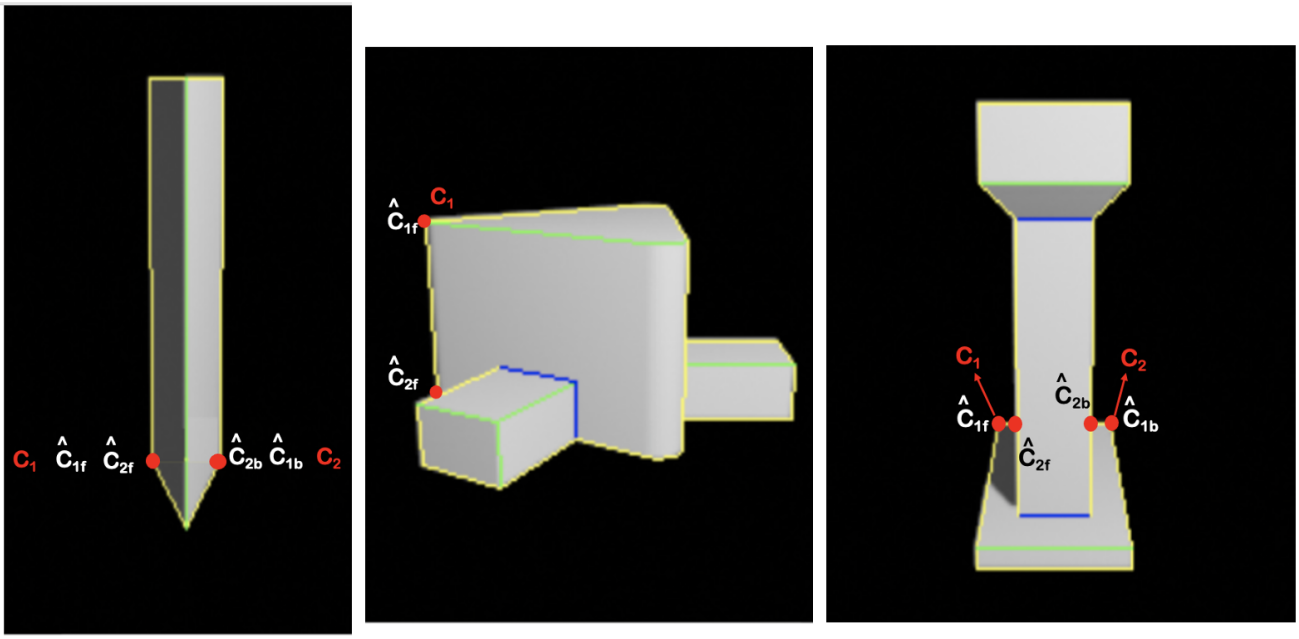

Figure 2. Examples of occlusion cases. In each case a contour C, with endpoints c_1 and c_2, is occluded. We mark $\hat{c_{1f}}$ and $\hat{c_{2f}}$, the first visible point and last visible point in the forward search respectively. We also mark $\hat{c_{1b}}$ and $\hat{c_{2b}}$, the first visible point and last visible point in the backward search respectively.

5. Line Labelled Groundtruths

Dataset Sampling Details We render the images in Blender, at a fixed camera position of (x=10, y=2.5, z=0). The light source is set at position (x=10, y=7, z=5). Roughly $N = 10$ poses are chosen per model, sampled at equal frequency around each axis.

Additional Visualizations We show additional groundtruth labeled scenes in Figures 3 and 4. As is convention in our paper, in Fig. 3, we label the obscuring contours yellow, the concave contours blue, and the convex contours green. The results in Fig. 3 are after contour extraction and classification from Section 3.2 and 3.3 respectively, which deal with solely linear contours. As a result, we develop the grouping algorithm in Section 3.4, with more results shown in Fig. 4.

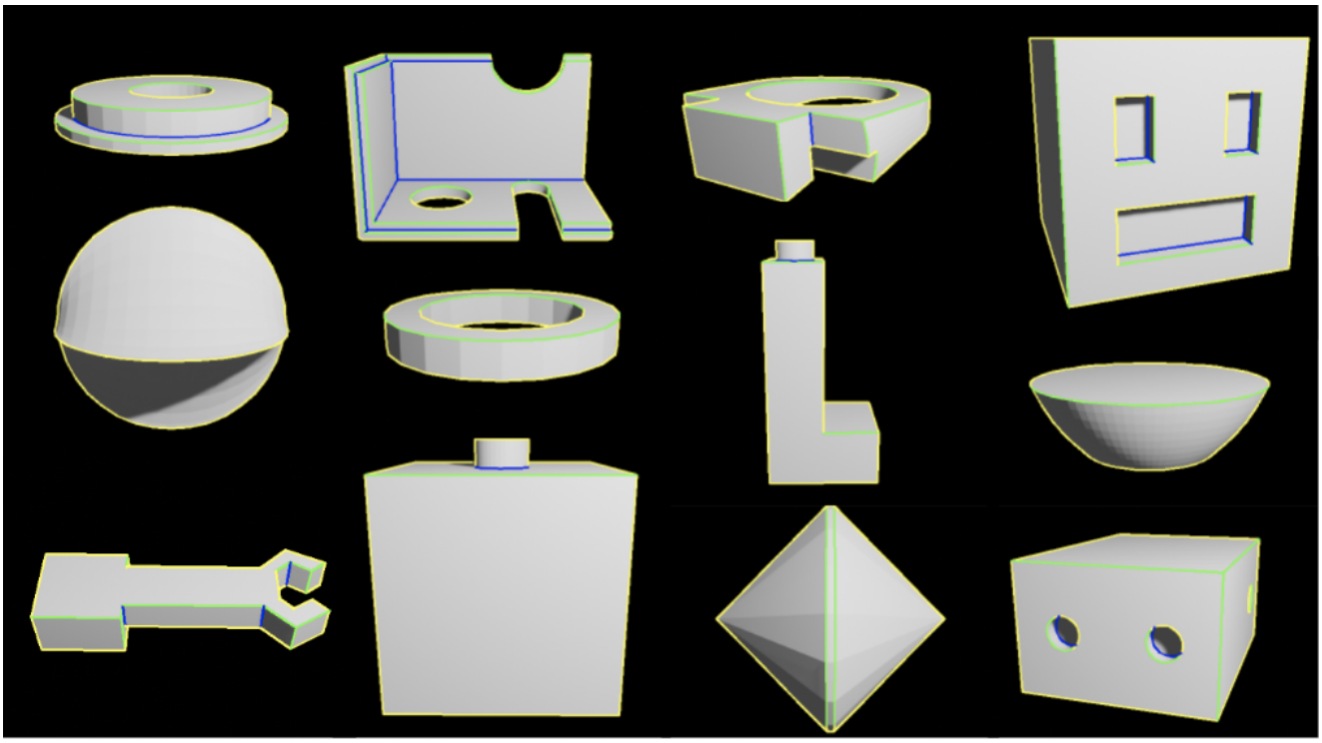

Figure 3. Examples of ground truths, with labels: obscuring (yellow), convex (green), concave (blue).

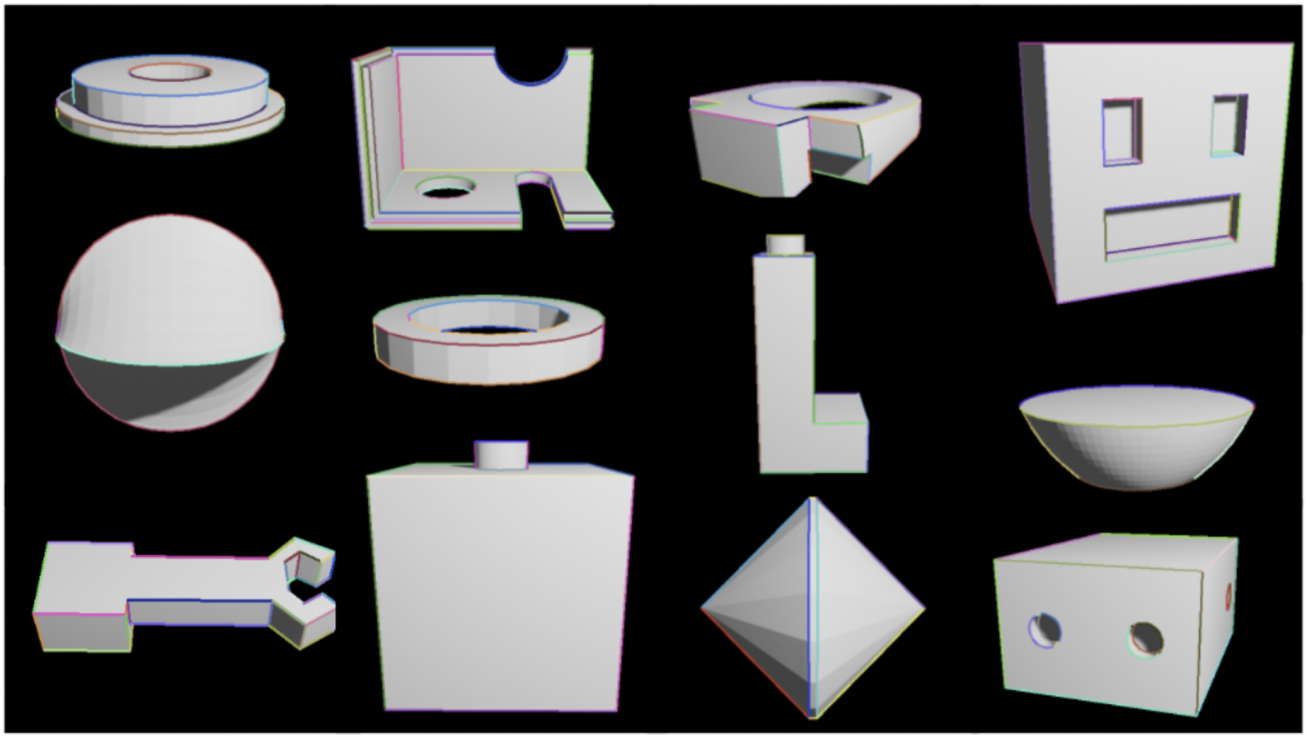

Figure 4. Examples of ground truths, with groupings: distinct colors represent unique groups.

Data: Already priority ordered heap H, where each element h contains a key, $h.key$, that indicates the contour, and a value, $h.value$, for the mean normal vector of the group. The priority is indicated in Section 3.4.
Result: list $final\_groups$ that contains final grouped contours

```
//initial grouping
while heap do
    h ← pop(heap); //pops top priority contour
    h_prev ← copy(h);
    for n in neighbors(h) do
        a ← angle(h.value, n.value);
        if a > C_thresh or label(h) ≠ label(n) then
            continue; //grouping condition fails
        end
        h.key ← new_contour(h.key, n.key);
        h.value ← new_mean(h.value, n.value);
    end
    if h_prev = h.key then
        groups ← append(h.key);
        // conclude grouping for this contour
    end
end
// further linking
for g in groups do
    for n in neighbors(g) do
        n1 ← length(super_neighbor_endpt(g, n));
        n2 ← length(super_neighbor_endpt(n, g));
        // super_neighbor_endpt returns neighbors
          for arg1 at endpoint closest to arg2
        a ← angle(normal(g), normal(n));
        cond1 ← (n1 = 1 and n2 = 1);
        cond2 ← (a < C_thresh2);
        cond3 ← (label(g) = label(n));
        if cond1 and cond2 and cond3 then
            g ← append(n)
            place(g) //correctly place g back in
              groups list
        else
            final_groups ← append(g)
        end
    end
end
return final_groups
```

Algorithm 2: Pseudocode for contour grouping, given ordered heap H, described in Section 3.4. We implement the initial grouping and further linking, from Section 3.4. C_{thresh} is 3 radians and $C_{thresh2}$ is 30 radians.

6. Contour Labelling

In this section, we describe the training details of our segmentation experiments in Section 4. Additionally, we provide more visualization results, similar to Fig 3 in the main report.

6.1. Training Details

We use the Detectron2[8] framework, and follow their base settings for both SOLOv2[7] and Mask2former[1]. The input image sizes of our dataset are 512X512.

For SOLOv2[7], we use stochastic gradient descent (SGD). The base learning rate is 0.005 with multistep learning rate scheduler at fixed steps and decay of 0.0001. Data augmentation is performed as flips, and scale sampling from image sizes [400,460,480,512]. Batchsize is 8 images.

For Mask2former[1], the optimizer is AdamW, with a step learning rate scheduler. The base learning rate is 0.0001 with decay of 0.9 and 0.95 fractions of total training iterations by factor of 10. For data augmentation, random scale sampling is performed from range 0.3 to 2.0, followed by fixed size crop to 512X512. For inference, we threshold detection scores with a threshold of 0.5. Batch size is 8 images. Note that for Mask2former + earlyfusion with Bézier data, we first generate Bézier data on shapes in an offline manner without the ImageNet background. We aim to see if using Bézier curve information on shapes only, has any direct effect. Results are reported in main paper. Lack of compute resources currently makes it infeasible to run Bézier detection online on images with random backgrounds during training. Examples of Bézier data are shown in Fig 5.

All models are trained until training convergence is reached.

6.2. Qualitative Results

In the main report, Fig 3 shows results on both seen and unseen objects. Here we show results on solely unseen objects, as we are very interested in domain transfer. We are motivated by these set of results, because in cases where there are no CAD models, use of deep learning segmentation architectures is highly important for contour extraction. We show our visualizations in Figures 6 and 7.

7. 3D Reconstruction

We use Adam optimizer, with base learning rate of 0.0001 and beta decays of 0.9, 0.999. The loss function we use is Binary Cross Entropy(BCE) Loss instead of Negative Log Likelihood which is used in the original paper 3DR2N2 [2]. In our experiments, this loss converges better and provides more stable training. The batch size ranges from 6 to 12, depending on input image and output voxel sizes. Training is performed until convergence.

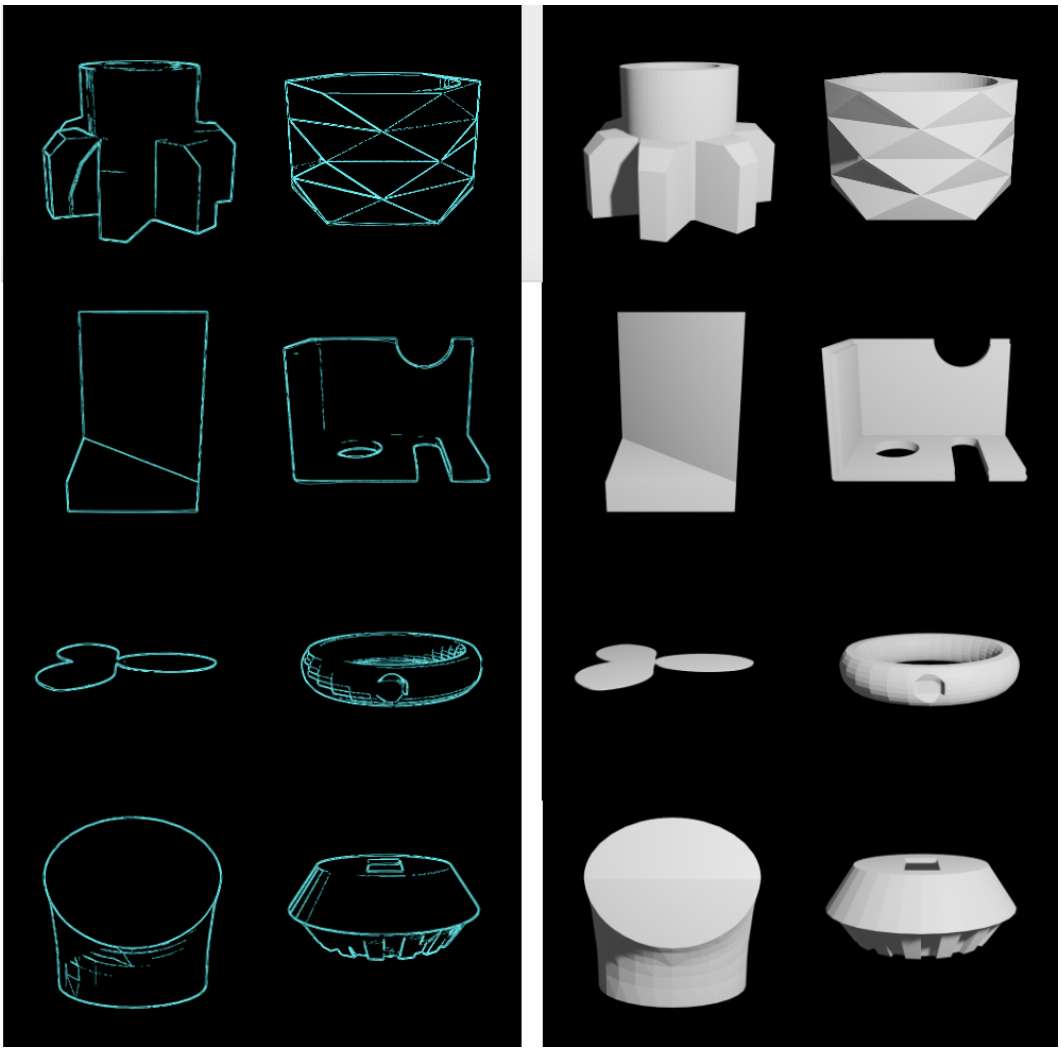

Figure 5. Example Bézier masks, processed on rendered images with no clutter, are shown in the left columns. The original images are shown in the right columns.

References

[1] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. 2022.

[2] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[3] Simon Christoph Stein, Markus Schoeler, Jeremie Papon, and Florentin Worgotter. Object partitioning using local convexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–311, 2014.

[4] M. B. Clowes. On seeing things. *Artif. Intell.*, 2:79–116, 1971.

[5] David A. Huffman. Realizable configurations of lines in pictures of polyhedrat. 2013.

[6] Jitendra Malik. Interpreting line drawings of curved objects. *International journal of computer vision*, 1(1):73–103, 1987.

[7] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020.

[8] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

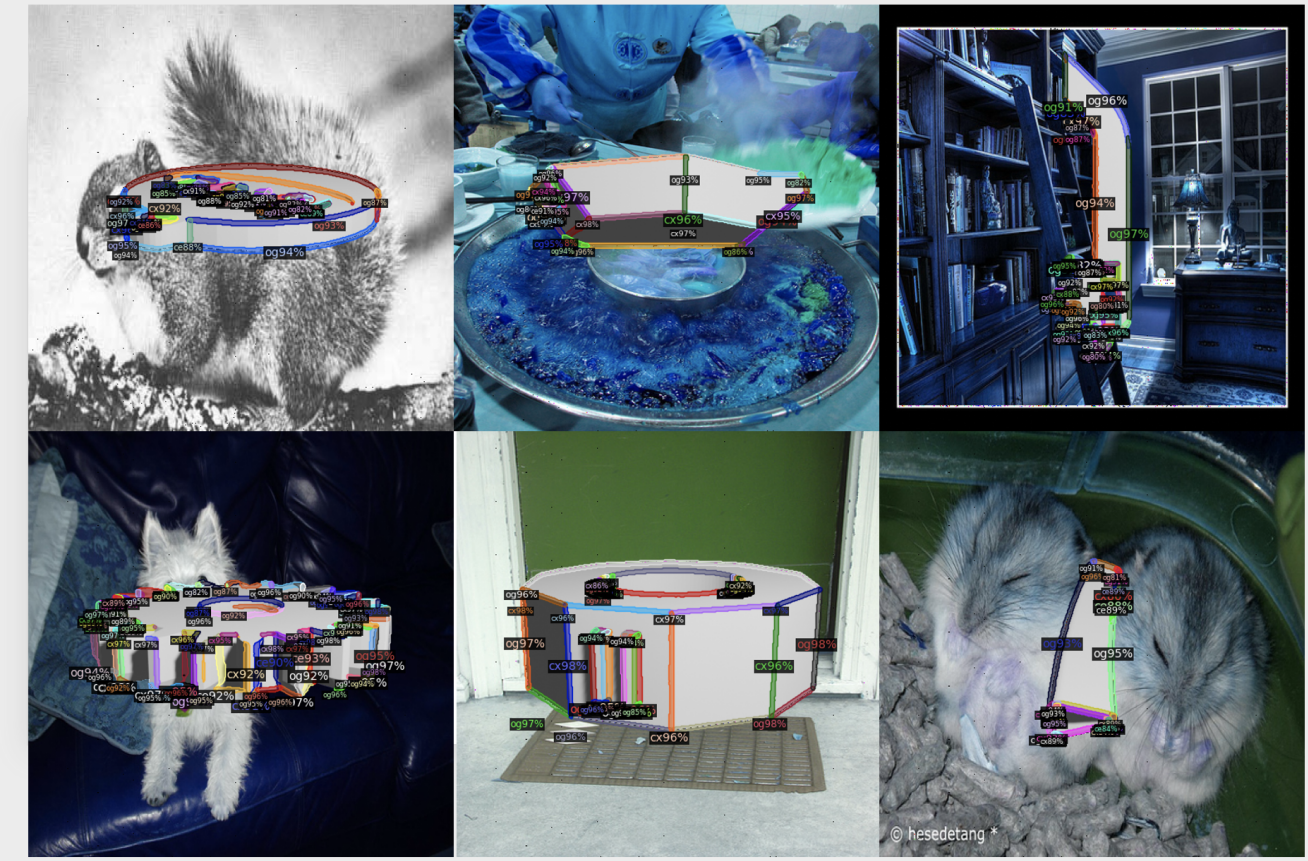

Figure 6. Poor Qualitative results for Mask2former + earlyfusion on unseen data.

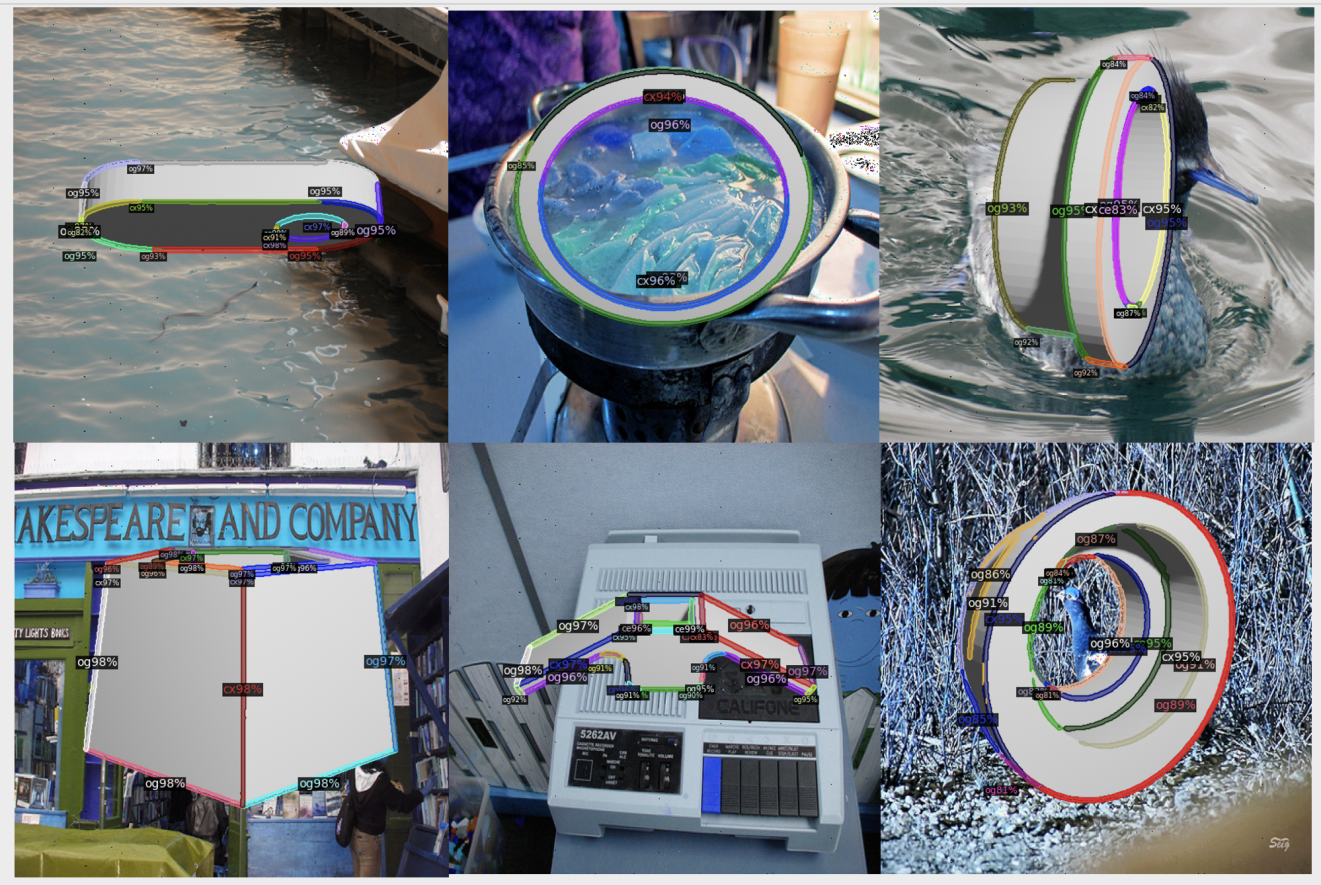

Figure 7. Good Qualitative results for Mask2former + earlyfusion on unseen data.